Seqalert—a daily sequence alertness server for the EMBL and SWISSPROT databases

Benny Shomer

EMBL Outstation—The European Bioinformatics Institute, The Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SD, UK

Received on March 31, 1997, revised and accepted on May 14, 1997

Abstract

Motivation: The aims were to: enable users to deposit complex search profiles against the sequence databases; interface to an independent Sequence Retrieval System (SRS) server through the network to perform these searches on a daily basis through the last day's updates of these databases; mail users the reformatted search results, enabling local usage when loaded by a WWW browser.

Results: The deposition of one to many search profiles by the user leads to a daily search of the EMBL and SWISSPROT databases. The search profile is restricted to entries that were deposited during the last 24 h by using the SRS query manager to combine search sets. If the search is successful, the resulting html page is modified from relative URLs to absolute ones, enabling local usage by loading from disk. The results are sent to the user by e-mail.

Availability: The programs are installed as a server at the URL: http://www.ebi.ac.uk/contribs/benny/alert/alert.html and are available as a package for installation as servers from the EBI ftp server at: ftp://ftp.ebi.ac.uk/pub/software/unix/seqalert/seqalert1.0.tar.gz. Installation as a server can also be done by sites that do not operate an SRS server.

Contact: E-mail: bshomer@ebi.ac.uk

Introduction

The main sequence databases for amino acids (SWISS-PROT) and nucleic acids (EMBL) are growing at extremely fast rates, which increase with the development of new sequencing technologies. Currently, the monthly average rate of new entry depositions into the EMBL/Genbank/DDBJ collaborative databases is 75,000 entries (EMBL Nucleotide Sequence Database, Release Notes, Appendix B—Database Growth Tables). This high rate makes it extremely difficult for scientists to keep track of the availability of new entries in these databases.

There are two main sources of information for the characterization of sequence data: sequence similarity searches and author contributed annotation. Seqalert was developed to automate textual searches of the data annotation section of the entries.

The Sequence Retrieval System (SRS) provides a powerful and complex means of searching through annotation in text-represented ("flatfile") entries (Etzold and Argos, 1993). SRS allows users to select various data fields to search through, based on the various tags in the flatfile. Users may combine search phrases in and between fields with logical operators, and use advanced features such as regular expressions to create exhaustive searches.

Seqalert was developed to enable users in remote locations to use a WWW interface to create and maintain various search profiles that match the structure and syntax of SRS search profiles. Users are identified by a combination of a user name and a password to provide privacy. The system allows users to create, modify or delete search profiles at any given moment. The search profiles of each user are fetched by the system on a daily basis. They are interpreted and sent to an SRS server. The SRS server does not need to reside on the site that provides the Seqalert service since the search is being done through the network (WWW) interface of SRS. The first search is performed in the 'Date' field for all the EMBL and SWISSPROT entries that were deposited in the last 24 h. The search profiles are then performed sequentially.

The system then uses the SRS query manager interface to combine each of the search profiles with the date search, resulting only with the profile matching entries from the last 24 h. If any entries are found, the system manipulates the URLs, which are embedded in the resulting html output, replacing the relative paths, which are generated by the server, with absolute ones which include the 'http' locator and the server's network identity. This enables users to save the mailed output to disk and load it by a WWW browser through the file opening mechanism. The embedded hyperlinks are then functional.

The SRS servers keep the search records of users for a limited time, depending on disk space constraints. Each search session is identified by a unique identifier that is embedded into the various hyperlinks in the output. Thus, if the users of Seqalert examine the output soon enough, their search sets still exist and they can select the query manager and continue the session that was initiated earlier by the Seqalert system.
Methodology

The system is based on three separate programs: seqalert, checkalerts and runalerts. All programs were written in the Python language (Lutz, 1996). Seqalert creates the WWW interface and allows users to manage their profiles. Checkalerts examines the profiles against the full releases of the databases and attempts to trap inefficient or wrong profiles. Runalerts interprets user profiles, submits searches to the SRS server, manipulates the results and sends them to the users.

Seqalert

**Interface.** The program interacts with the user through a WWW form which requests the user to enter the user name, password and status (new or veteran user). New users are also requested to provide their e-mail address. Once invoked, the program checks the user identity. If the user is new, the program creates a new user profiles file and displays a form that allows the user to create a profile. If the user exists, the program loads and displays the profiles and allows the user to edit or delete each of them, or to create new ones. When creating a new profile, a form is presented, which is very similar to the SRS search form for sequence data libraries, but without the need to select libraries or display format characteristics, and without the date field. The same form is presented, pre-loaded with the profile contents, when editing a profile.

**Implementation.** The program uses two classes: a profile class and a profilemanager class. The profile class has methods for management of a singular profile, loading and dumping its information and representing it in text, html or a python syntax format. The latter format enables python to read profile lines from a file and execute them dynamically into memory without the need for interpretation. The profilemanager class is responsible for all the manipulations on profiles of a single user. It has methods for loading and saving a set of profiles. A display method implements the representation of the profile as an html document that contains form interfaces to the methods for creating a new profile (one form for the whole display) and for editing or deletion of a profile (associated with each profile). Each profile is identified by a unique profile name that can be selected by the user. When invoked by a user call, the main program identifies the user and validates the identity and password. It then invokes a profilemanager instance, providing it with the user name and the URL that should be embedded as the 'action' variable of the output <form> tag. The manager reads the information in the profiles file and for each profile it invokes a profile instance. User requests (create, edit, delete) are executed on these instances and a new image of the profiles is saved to disk.

For the production of HTML code, seqalert uses a module WWW, which defines an HTML generating class and has been described elsewhere (Shomer, 1997).

Runalerts

This program is written for invocation mainly as a Unix cron job, intended to be executed on a daily basis. It obtains all the user names, which have been deposited, and performs a profile search for each user. It uses a class MHTTP that inherits python's standard HTTP class for operations associated with the httpd protocol (Fielding et al., 1996). MHTTP extends HTTP by allowing an easier invocation of requests from httpd daemons, including submission of form queries with the POST method, and interpretation of resulting server output. Almost all the program's operations are enclosed in a 'try' clause (python's exception handling call) where failures raise descriptive error messages. It enables the cron manager to trap the error and mail it to the service operator.

**Profile searches.** The program loads from the user's file the user's e-mail address and all the profiles (see Figure 1). It then communicates with the SRS server to obtain a user Id. SRS uses the Id to allocate a temporary storage area on the server that allows the keeping track of user queries. Next, the program performs a first query, which requests all the entries released to the EMNEW and SWISSNEW databases in the past 24 h, by using the 'date' field with the appropriate date calculated. This causes SRS to register this query as 'Q1'. The program then counts the number of queries in the profiles, submits all of them and keeps track of the success rate. If a query resulted in no hits, the total number of queries per user needs to be decreased, since this query is not registered with the query manager. For each of the successful queries, the program then invokes the query manager, requesting 'Q1' to be combined with the successful query, which results in filtering of the results to leave only the new ones. If this search results in hits, the program interprets the html output and replaces all the relative URLs, which were produced by SRS, with absolute ones, which include the server address. This is done to enable the user to use the form by loading it locally into the browser. The program then uses a class that handles e-mail to send the results to the user, accompanied by explanations on how to use the output.

Checkalerts

Checkalerts is very similar to runalerts in its structure and function, and is also written to be invoked as a cron job. It maintains a record of users and the date stamp of their profile files. For each new or modified profiles file, the program checks the user profiles against the full releases of EMBL and SWISSPROT. It then scans the results. The program informs the user by e-mail if no results were obtained or the server informs of a syntax error. It also refers the user to a help document with examples for search profiles.
The ability to change the profiles at any given moment is gaurdable through a configuration file. Methods or information, to keep track of new sequencing results may help any scientist, who uses molecular biology sequences without automation. The system is flexible, allowing one to operate the service from many different servers, regardless of the fact that the site which provides the service may not operate an SRS server. The system is easily configurable through a configuration file.

The alertness service depends on sequence annotation. The sequence alertness system will contribute to the awareness level of scientists to new protein and nucleic acid sequences of interest, reducing time and resource requirements, helping to overcome the problem of managing the constantly increasing amounts of information in the databases.

Discussion

An alertness system for nucleic acid and protein sequences, based on the sequence annotation, has been described. Such a service may help any scientist, who uses molecular biology methods or information, to keep track of new sequencing results. The current rate of depositions to the sequence databases is so high that it is impractical to stay alert to new sequences without automation. The system is flexible, allowing one to operate the service from many different servers, regardless of the fact that the site which provides the service may not operate an SRS server. The system is easily configurable through a configuration file.

The alertness service depends on sequence annotation. The same limitation exists in any searches (such as normal SRS) which are text based. Nevertheless, basing searches on annotation is of great value since most of the annotation normally comes from the sequence depositor and is likely to be accurate many times. Seqalert does not perform homology searches against a user’s probe sequence. Such a system has, however, recently been developed (Basset et al., 1995).

The Seqalert program provides a clear and easy interface for the manipulation of search profiles through the WWW. The ability to change the profiles at any given moment allows users to experiment with the system until optimal results are obtained. Since the program ignores empty profiles, users do not need specifically to request their removal from the system, avoiding inconvenience. The profiles interface is accompanied by a detailed document, which provides precise instructions on composing searches, providing many examples for the syntax and logic behind the queries.

The program uses the query manager for two main reasons. First, using the query manager enables the program to allocate all four available search fields for the user’s profile, allowing the creation of highly complex queries. Second, it enables searching through the ‘date’ field indexes only once per user, instead of once per profile, saving system resources.

The step of requesting a user Id from SRS must be performed separately for each user to avoid a situation in which users use the query manager to view other search profiles, resulting in loss of confidentiality.

The system is currently tuned to communicate with releases 4.06 and 4.08 of SRS since these are the releases that are actually implemented on all the major servers. A new release of runalerts, which will be tuned to communicate with release 5.0 of SRS, is currently in preparation and is likely to be ready before SRS 5.0 will be widely used.

An important factor that must be considered by operators of the service is the quality of the SRS site to which the runalerts program is directed. The site must be well maintained with daily regular updates, and all the index files should be accessible to the SRS program. This quality factor varies between the currently available sites.

The sequence alertness system will contribute to the awareness level of scientists to new protein and nucleic acid sequences of interest, reducing time and resource requirements, helping to overcome the problem of managing the constantly increasing amounts of information in the databases.

References